Distributed Weighted Vertex Cover via Maximal Matchings

FABRIZIO GRANDONI
Università di Roma Tor Vergata
JOCHEN KÖNEMANN
University of Waterloo
and
ALESSANDRO PANCONESI
Sapienza Università di Roma

In this paper we consider the problem of computing a minimum-weight vertex-cover in an n-node, weighted, undirected graph $G = (V, E)$. We present a fully distributed algorithm for computing vertex covers of weight at most twice the optimum, in the case of integer weights. Our algorithm runs in an expected number of $O(\log n + \log \hat{W})$ communication rounds, where $\hat{W}$ is the average vertex-weight. The previous best algorithm for this problem requires $O(\log n(\log n + \log \hat{W}))$ rounds and it is not fully distributed.

For a maximal matching M in G it is a well-known fact that any vertex-cover in G needs to have at least $|M|$ vertices. Our algorithm is based on a generalization of this combinatorial lower-bound to the weighted setting.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Approximation algorithms, distributed algorithms, maximal matching, vertex cover

1. INTRODUCTION

We are given an undirected graph $G = (V, E)$ and non-negative integer vertex weights w_v for all vertices $v \in V$. A *vertex cover* is a subset $C \subseteq V$ such that each edge $e \in E$ has at least one end-point in C. In the *minimum-weight vertex-cover* problem we want to compute a vertex-cover of smallest total weight.

A preliminary version of this paper appeared in COCOON'05.
This work was carried out while the first two authors were at Sapienza Università di Roma.
Authors' addresses: F. Grandoni, Dipartimento di Informatica, Sistemi e Produzione, Università di Roma Tor Vergata, Via del Politecnico 1, 00133, Roma, Italy, email: grandoni@disp.uniroma2.it; J. Könemann, Department of Combinatorics and Optimization, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada, email: jochen@uwaterloo.ca; A. Panconesi, Dipartimento di Informatica, Sapienza Università di Roma, Via Salaria 113, 00198, Roma, Italy, email: ale@di.uniroma1.it.

Computing minimum-weight vertex-covers is NP-hard [Garey and Johnson 1979]. Papadimitriou and Yannakakis [1991] show that the problem is APX-hard. Recently, Dinur and Safra [2002] showed that it is NP-hard to approximate the vertex-cover problem to within any factor smaller than $10\sqrt{5} - 21 > 1.36$, improving on the previous best 7/6 lower bound by Håstad [2001].

On the positive side, the best known approximation algorithm for the vertex-cover problem is due to Karakostas [2005] who presented a $(2 - \Theta(1/\sqrt{\log(n)})$ approximation for the problem. This improves upon earlier $(2-o(1))$ approximation algorithms due to Bar-Yehuda and Even [1981], Hochbaum [1982], Monien and Speckenmeyer [1985], and Halperin [2002].

In the distributed setting, it is known how to compute a 2-approximate vertex cover in the unweighted case. This can be achieved by computing a maximal matching in the graph and by including the matched nodes in the cover. A maximal matching can be computed in $\mathrm{O}(\log^4 n)$ rounds via the algorithm of Hanckowiack et al. [2001], and in $\mathrm{O}(\Delta + \log^* n)$ rounds via the algorithm of Panconesi and Rizzi [2001]. Both algorithms are deterministic. Maximal matchings can also be computed in an expected number of $\mathrm{O}(\log n)$ rounds via the randomized algorithm of Israeli and Itai [1986].

For the weighted case a $(2+\epsilon)$-approximation can be computed deterministically in $\mathrm{O}(\log n \log \frac{1}{\epsilon})$ many rounds by using the algorithm of Khuller et al. [1994]. Their algorithm is stated as a PRAM algorithm, but it is readily seen to be a bona fide distributed algorithm. Let $\hat{W}$ be the average weight. Then, by setting $\epsilon = 1/(n\hat{W} + 1)$, the latter algorithm computes a 2-approximate vertex cover in $\mathrm{O}(\log n(\log n + \log \hat{W}))$ communication rounds. Note that the above choice of ϵ requires global knowledge of the quantity $n\hat{W}$. This assumption may not be realistic in all scenarios.

In this paper we present an improved fully-distributed algorithm to compute a 2-approximate weighted vertex cover. Our main result can be stated as follows. Let W and Δ denote the largest weight and degree of a node, respectively. We recall that $\hat{W}$ is the average weight of a node.

THEOREM 1.1. *There is a fully distributed algorithm which computes a* 2*-approximate weighted vertex cover in an expected number of* $\mathrm{O}(\log n + \log \hat{W})$ *communication rounds. The message size is* $\mathrm{O}(\log W)$ *and the local computation done in each round is* $\mathrm{O}(\Delta \log(\Delta W))$ *in expectation.*

For a maximal matching M in G it is a well-known fact that any vertex cover in G needs to have at least $|M|$ vertices. Our algorithm is based on a generalization of this property to the weighted setting.

The basic idea is to expand each node v of weight w_v into w_v *micro-nodes* $v(1), v(2) \ldots, v(w_v)$, and connect each $v(i)$ to every $u(j)$ whenever vu is an edge of the network. Then a maximal matching in the auxiliary graph is computed. The vertex cover is given by the nodes for which all corresponding micro-nodes are matched. If the maximal matching is computed via the fully-distributed algorithm of Israeli and Itai, the algorithm halts in an expected number of $\mathrm{O}(\log n + \log \hat{W})$ rounds.

A naive implementation of the matching algorithm by Israeli and Itai leads to pseudo-polynomial message and time complexity in each round. The main insight

leading to the bounds on message-size and local computation time in Theorem 1.1 is to keep an implicit representation of the auxiliary graph and a maximal matching in it.

The rest of this paper is organized as follows. In Section 2 we introduce some preliminaries. Our algorithm relies on a careful adaptation of the matching algorithm by Israeli and Itai. We present this adaptation in Section 3. Finally, Sections 4 and 5 deal with the naive and refined versions of our weighted vertex cover algorithm, respectively.

2. PRELIMINARIES

The minimum-weight vertex cover problem can be formulated as an *integer linear program* (ILP):

$$\begin{array}{ll} \min & \sum_{v\in V} w_v x_v \\ \text{s.t.} & \\ x_v + x_u \geq 1, & \forall vu \in E; \\ x_v \in \{0,1\}, & \forall v \in V. \end{array}$$

Each assignment of the variables which satisfies the constraints (*feasible solution*) corresponds to a vertex cover containing exactly the nodes v with $x_v = 1$. By (LP) we denote the natural *linear programming relaxation* of (ILP).

Let $N(v)$ be the set of neighbors of v. The *linear programming dual* (D) of (LP) is:

$$\begin{array}{ll} \max & \sum_{vu\in E} y_{vu} \\ \text{s.t.} & \\ \sum_{u\in N(v)} y_{vu} \leq w_v, & \forall v \in V; \\ y_{vu} \geq 0, & \forall vu \in E. \end{array}$$

By *weak duality* (e.g., see [Chvátal 1983]), the value of each feasible solution of (D) is a lower bound for the value of every feasible solution of (LP) and hence (ILP).

In this paper we consider the standard synchronous message-passing model of computation. The computation proceeds in rounds. In each round, a node can send/receive a message (of unbounded size) to/from each one of its neighbors, and execute an unbounded amount of computation. No global knowledge is available (including the number n of nodes in the graph). The algorithms presented can be easily modified so as to work in a (non-faulty) asynchronous system also.

We use $B(p)$, $p \in [0,1]$, to denote a *Bernoulli* random variable, which takes value 1 with probability p and 0 otherwise. A *random bit* is a Bernoulli variable $B(0.5)$.

3. DISTRIBUTED MAXIMAL MATCHING

A *matching* of a graph $G = (V, E)$ is a subset $M \subseteq E$ such that no two edges of M are incident to the same node. The results of the next sections are based on the following simplified version $\mathcal{M}$ of the distributed maximal-matching algorithm of Israeli and Itai [Israeli and Itai 1986].

Algorithm $\mathcal{M}$ works in phases, each one consisting of a constant number of rounds. In each phase, a matching is computed and the edges incident on matched nodes are removed. The algorithm halts when no edge is left. The maximal matching is given by the union of the matchings found in the different phases.

In a given phase a matching is computed in the following way. Let $G' = (V', E')$ be the current graph. By $N'(v)$ and δ'_v we denote the set of neighbors of v and the degree of v in G', respectively. Each node v randomly decides to be a *sender* or a *receiver* with probability one half. Note that the same node may play a different role in different phases. Each sender u selects one neighbor $v \in N'(u)$ uniformly at random and makes a *proposal* to v. Each receiver v which receives at least one proposal, selects one of the proponents (arbitrarily) and *accepts* its proposal. The matching is given by the edges corresponding to accepted proposals.

Let a node v be *good* if at least one third of its neighbors u have degree $\delta_u \leq \delta_v$. To prove the bound on the number of rounds, we use the following simple combinatorial result [Israeli and Itai 1986]:

LEMMA 3.1. *At least one half of the edges of a graph are incident to good nodes.*

THEOREM 3.2. *Algorithm $\mathcal{M}$ computes a maximal matching in $\mathrm{O}(\log n)$ expected rounds.*

PROOF. The correctness of the algorithm is trivial.

We show that in each phase at least a constant fraction of the edges is removed in expectation. This implies that the expected number of rounds is $\mathrm{O}(\log(n^2)) = \mathrm{O}(\log n)$. Consider a good node v of G' in a given phase. The probability P'_v that v accepts a proposal is lower bounded by:

$$P'_v \geq \frac{1}{2}\left(1 - \prod_{u \in N'(v)}\left(1 - \frac{1}{2\delta'_u}\right)\right).$$

From the definition of good nodes:

$$\prod_{u \in N'(v)}\left(1 - \frac{1}{2\delta'_u}\right) \leq \prod_{u \in N'(v):\delta'_u \leq \delta'_v}\left(1 - \frac{1}{2\delta'_v}\right) \leq \left(1 - \frac{1}{2\delta'_v}\right)^{\frac{\delta'_v}{3}} \leq e^{-\frac{1}{6}}.$$

Thus $P'_v \geq (1 - e^{-1/6})/2$. Hence, by Lemma 3.1, the expected number of edges removed is at least a fraction $(1 - e^{-1/6})/4$ of the total. □

4. DISTRIBUTED VERTEX COVER VIA MAXIMAL MATCHINGS

In this section we show how the problem of computing an approximate vertex cover can be reduced to that of computing a maximal matching in an auxiliary graph. Using this reduction, we show how to compute a 2-approximate vertex cover in $\mathrm{O}(\log n + \log \hat{W})$ expected rounds via algorithm $\mathcal{M}$ of section 3.

Consider the following auxiliary graph $\widetilde{G}$. For each node v of G, $\widetilde{G}$ contains w_v *micro-nodes* $v(1), v(2) \dots v(w_v)$. Two micro-nodes $v(i)$ and $u(j)$ are adjacent if and only if vu is an edge of G. In Figure 1 an example of the reduction is given.

Let M be a maximal matching in $\widetilde{G}$. By $V(M)$ we denote the set of nodes v of G such that all the corresponding micro-nodes $v(i)$ are matched by M.

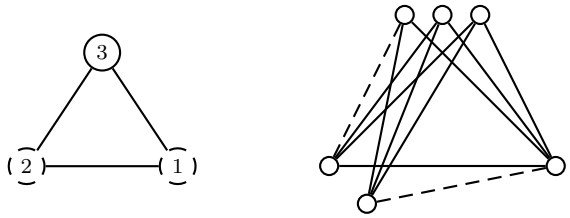

Fig. 1. A weighted graph G (on the left) with the corresponding auxiliary graph $\widetilde{G}$. A maximal matching M of $\widetilde{G}$ is indicated via dashed lines. The dashed nodes of G form a vertex cover.

LEMMA 4.1. *Set $V(M)$ is a 2-approximate vertex cover of G.*

PROOF. Assume by contradiction that $V(M)$ is not a vertex cover. Thus there are two adjacent nodes v and u in G which do not belong to $V(M)$. This implies that there are two adjacent micro-nodes $v(i)$ and $u(j)$ in $\widetilde{G}$ which are not matched by M. Then the set $M' = M \cup \{v(i)u(j)\}$ is a matching, which contradicts the maximality of M.

Let `apx` and `opt` denote the weight of the vertex cover found and that of a minimum weight vertex cover, respectively. Moreover, let z_v be the number of micro-nodes in $\{v(1), v(2) \dots v(w_v)\}$ that are matched by M. A feasible solution of (D) is obtained by assigning to each dual variable y_{vu} the number of edges of the kind $v(i)u(j) \in M$. This solution is feasible since, for every $v \in V$:

$$\sum_{u \in N(v)} y_{vu} = z_v \leq w_v.$$

By weak duality we obtain

$$apx \leq \sum_{v \in V} z_v \leq 2 \sum_{vu \in E} y_{vu} \leq 2\, opt$$

and hence $V(M)$ is 2-approximate. □

Lemma 4.1 suggests a strategy to compute a 2-approximate vertex cover distributively. The idea is to simulate the behavior of algorithm $\mathcal{M}$ on a virtual auxiliary graph $\widetilde{G}$, and then to select the nodes in the vertex cover as suggested by Lemma 4.1.

Specifically, each node simulates the execution of the algorithm on the corresponding micro-nodes $v(i)$ in $\widetilde{G}$. Whenever two micro-nodes $v(i)$ and $u(j)$ of $\widetilde{G}$ need to communicate, nodes v and u are responsible for allowing such communication. The vertex cover is given by the nodes v such that all the corresponding micro-nodes $v(i)$ are matched by the maximal matching computed.

Since the virtual auxiliary graph contains $\mathrm{O}(n\hat{W})$ nodes, the total number of rounds is $\mathrm{O}(\log(n\hat{W})) = \mathrm{O}(\log n + \log \hat{W})$.

This naive application of Lemma 4.1 has two major drawbacks. The first problem

is the large message size. In fact, in each phase all the (remaining) micro-nodes of v may send a proposal to some micro-node of u. Thus the message size is $\Omega(W)$.

A second problem is the time complexity of the algorithm: consider a node v in a given phase. Each micro-node $v(i)$ of v, with probability one half, needs to select one neighbor out of $\Theta(\Delta W)$ uniformly at random. This random selection can be performed in $\Theta(\log(\Delta W))$ expected time, assuming that the cost of generating a random bit is O(1) (e.g., see [Cormen et al. 1992]). Thus the expected time complexity of each phase is $\Omega(W \log(\Delta W))$.

In next section we show how to solve both problems by creating the matchings *implicitly*.

5. AN IMPROVED ALGORITHM

In this section we present an improved fully distributed algorithm $\mathcal{A}$ for computing a 2-approximate vertex cover. Algorithm $\mathcal{A}$ still requires $O(\log n + \log \hat{W})$ expected rounds, but it reduces the size of the messages to $O(\log W)$ and the expected time complexity of each phase to $O(\Delta \log(\Delta W))$.

The basic structure of algorithm $\mathcal{A}$ is analogous to the structure of the naive algorithm described in previous section: in each phase, a matching in the current auxiliary graph $\widetilde{G}'$ is computed, and the matched nodes are removed from $\widetilde{G}'$ (together with all the edges incident to them). The algorithm halts when no edge is left. The vertex cover is given by the nodes v such that all the corresponding micro-nodes $v(i)$ are matched by one of the matchings computed.

The main novelty in Algorithm $\mathcal{A}$ is that matchings are created *implicitly*: in each phase each node only knows the number of the corresponding matched micro-nodes. Intuitively, this simplification is allowed by the symmetry properties of $\widetilde{G}$: all the remaining micro-nodes corresponding to a node v have the same degree and share the same neighborhood. This invariant is kept by all the induced subgraphs of $\widetilde{G}$.

Algorithm $\mathcal{A}$, which is described in Figure 2, works in phases. Each phase consists of a constant number of communication rounds. Each node v has an associated *state* s_v, which is initially *active*. In each phase, some of the *active* nodes switch to the state *inside* or *outside*, and the algorithm terminates when no *active* node is left. When a node leaves the *active* state, it halts. At the end of the algorithm, the *inside* nodes form a vertex cover.

In more details, each node v has an associated *residual weight* w'_v, which is initially w_v. The residual weight w'_v can be interpreted as the number of micro-nodes $v(i)$ of v in the current auxiliary graph $\widetilde{G}'$. Note that all the micro-nodes $v(i)$ have the same degree W'_v:

$$W'_v = \sum_{u \in N(v)} w'_u.$$

In each phase, the expected residual weight of active nodes decreases. The decrease of w'_v in a given phase reflects the number of micro-nodes of v that have been matched in that phase.

At the beginning of each phase, each *active* node v sends w'_v to all its currently active neighbors $N'(v)$. The neighbors with $w'_u = 0$ are removed from $N'(v)$. If

```
w'_v = w_v; N'(v) = N(v), s_v = active;
while (s_v = active) {
   send w'_v and receive w'_u to/ from all u ∈ N'(v);
   N'(v) = {u ∈ N(v) : w'_u > 0};
   if (|N'(v)| = 0)
      s_v = outside;
   else {
      compute the proposals p_v(u);
      send p_v(u) and receive p_u(v) to/ from all u ∈ N'(v);
      compute the counter-proposals c_v(u);
      send c_v(u) and receive c_u(v) to/ from all u ∈ N'(v);
      for (all u ∈ N'(v))
         w'_v = w'_v − c_v(u) − c_u(v);
      if (w'_v = 0) {
         send w'_v to all u ∈ N'(v);
         s_v = inside;
      }
   }
}
```

Fig. 2. Protocol for node v for 2-approximate vertex cover.

$N'(v)$ becomes empty, node v switches to the *outside* state. In fact, in this case the degree W'_v of the micro-nodes $v(i)$ is zero, and thus they will never be matched.

Otherwise, v sends a *proposal* $p_v(u)$ to each *active* neighbor $u \in N'(v)$. The value of $p_v(u)$ can be interpreted as the number of proposals directed from the micro-nodes of v to the micro-nodes of u. Let p'_v be the sum of the proposals $p_v(u)$:

$$p'_v = \sum_{u \in N'(v)} p_v(u).$$

This quantity can be viewed as the number of *micro-senders* among $v(1)$, $v(2)$,..., $v(w_v)$. We postpone a detailed description of how proposals are fixed until later.

For each proposal $p_u(v)$ received, node v replies with a *counter-proposal* $c_v(u)$. The counter-proposal $c_v(u)$ can be interpreted as the number of micro-nodes of v which accept proposals of micro-nodes of u. Let $c'_v = w'_v - p'_v$ be the number of *micro-receivers* of v. The sum of the counter-proposals for node v then needs to be at most c'_v (there cannot be more accepted proposals than *micro-receivers* in v). At the same time each counter-proposal $c_v(u)$ must not exceed the corresponding proposal $p_u(v)$ (*micro-receivers* of v cannot accept more proposals from the *micro-senders* of u than the proposals actually received). Given these restrictions, we choose a feasible set of counter-proposals $\{c_v(u)\}_{u \in N'(v)}$ arbitrarily, such that their sum is maximum. For example, let $N'(v) = \{u_1, u_2, \ldots, u_{|N'(v)|}\}$. For increasing values of i, $i = 1, 2, \ldots, |N'(v)|$, we can set $c_v(u_i) = \min\{p_u(v), c'_v - \sum_{j=1}^{i-1} c_v(u_j)\}$. Observe that, at the end of the process,

$$\sum_{u \in N'(v)} c_v(u) = \min\{c'_v, \sum_{u \in N'(v)} p_u(v)\}.$$

Eventually, node v decrements w'_v by the sum of all the counter-proposals $c_v(u)$

and $c_u(v)$ which have been sent and received by v, respectively:

$$w'_v = w'_v - \sum_{u \in N'(v)} (c_v(u) + c_u(v)) .$$

This decrement reflects the number of micro-nodes of v which are matched in the considered phase.

If w'_v becomes zero, node v sends w'_v to all its neighbors (for the last time) and switches to the *inside* state (since all the corresponding micro-nodes are matched).

Observe that the value of any feasible proposal (and thus of any counter-proposal) is at most W. Hence, the message size decreases to $O(\log W)$. It remains to show how to compute the proposals efficiently, without increasing the expected number of communication rounds.

A feasible set of proposals for a given node v is obviously obtained by simulating Algorithm $\mathcal{M}$: the proposals $p_v(u)$ are initially set to zero. Then, for w'_v times, an active neighbor $u \in N'(v)$ is selected at random with probability proportional to w'_u, and the corresponding proposal $p_v(u)$ is incremented by one with probability one half. Note that each $p_v(u)$, considered separately, is the sum of w'_v i.i.d. Bernoulli variables $B(\frac{w'_u}{2W'_v})$:

$$p_v(u) = \sum_{i=1}^{w'_v} B\left(\frac{w'_u}{2W'_v}\right). \tag{1}$$

However, this approach is too expensive for large value of w'_v. The idea is then to approximate the behavior of the proposals above. A natural choice is as follows:

$$p_v(u) = \left\lfloor \frac{w'_v w'_u}{2W'_v} \right\rfloor + B\left(\frac{w'_v w'_u}{2W'_v} - \left\lfloor \frac{w'_v w'_u}{2W'_v} \right\rfloor\right). \tag{2}$$

We observe that, with both (1) and (2), $E[p_v(u)] = \frac{w'_v w'_u}{2W'_v}$. However, the computation of (2) is faster.

There is a technical problem: for the proposals to be feasible, it must be (deterministically), $p'_v := \sum_{u \in N'(v)} p_v(u) \leq w'_v$ (there cannot be more *micro-senders* than *micro-nodes*). This is guaranteed for $w'_v \geq 2\delta'_v$, where $\delta'_v = |N'(v)| \geq 1$ is the number of currently *active* neighbors of v:

$$\sum_{u \in N'(v)} p_v(u) \leq \sum_{u \in N'(v)} \left(\frac{w'_v w'_u}{2W'_v} + 1\right) = \frac{w'_v}{2} + \delta'_v \leq w'_v.$$

For $w'_v < 2\delta'_v \leq 2\Delta$, we just simulate Algorithm $\mathcal{M}$ as described before.

Summarizing, proposals are set in the following way:

$$p_v(u) = \begin{cases} \sum_{i=1}^{w'_v} B\left(\frac{w'_u}{2W'_v}\right) & \text{if } w'_v < 2\delta'_v; \\ \left\lfloor \frac{w'_v w'_u}{2W'_v} \right\rfloor + B\left(\frac{w'_v w'_u}{2W'_v} - \left\lfloor \frac{w'_v w'_u}{2W'_v} \right\rfloor\right) & \text{otherwise.} \end{cases}$$

The following technical property of the proposals will be useful in later parts of the analysis.

LEMMA 5.1. *For fixed $\{w'_u\}_{u\in V}$, and for any two given active nodes v and $u \in N'(v)$,*

$$E_{u,v} := E\left[\left(1-\frac{1}{w'_v}\right)^{p_u(v)}\right] \leq e^{-\frac{w'_u}{4W'_u}}.$$

PROOF. Recall that, for $x \geq 1$, $(1-\frac{1}{x})^x \leq e^{-1}$ and $\lfloor x \rfloor \geq x/2$. We also remark that $W'_u \geq w'_v \geq 1$. If $w'_u < 2\delta'_u$,

$$E_{u,v} = E\left[\left(1-\frac{1}{w'_v}\right)^{\sum_{i=1}^{w'_u} B\left(\frac{w'_v}{2W'_u}\right)}\right] = \left(E\left[\left(1-\frac{1}{w'_v}\right)^{B\left(\frac{w'_v}{2W'_u}\right)}\right]\right)^{w'_u}$$
$$= \left(1-\frac{w'_v}{2W'_u}+\frac{w'_v}{2W'_u}\left(1-\frac{1}{w'_v}\right)\right)^{w'_u} = \left(1-\frac{1}{2W'_u}\right)^{w'_u} \leq e^{-\frac{w'_u}{2W'_u}}.$$

Consider now the case $w'_u \geq 2\delta'_u$. We further distinguish two subcases. If $\frac{w'_u w'_v}{2W'_u} \geq 1$,

$$E_{u,v} = E\left[\left(1-\frac{1}{w'_v}\right)^{\lfloor \frac{w'_u w'_v}{2W'_u}\rfloor + B\left(\frac{w'_v w'_u}{2W'_v} - \left\lfloor \frac{w'_v w'_u}{2W'_v}\right\rfloor\right)}\right] \leq E\left[\left(1-\frac{1}{w'_v}\right)^{\lfloor \frac{w'_u w'_v}{2W'_u}\rfloor}\right]$$
$$= \left(1-\frac{1}{w'_v}\right)^{\lfloor \frac{w'_u w'_v}{2W'_u}\rfloor} \leq \left(1-\frac{1}{w'_v}\right)^{\frac{w'_u w'_v}{4W'_u}} \leq e^{-\frac{w'_u}{4W'_u}}.$$

Otherwise ($\frac{w'_u w'_v}{2W'_u} < 1$):

$$E_{u,v} = E\left[\left(1-\frac{1}{w'_v}\right)^{\lfloor \frac{w'_u w'_v}{2W'_u}\rfloor + B\left(\frac{w'_v w'_u}{2W'_v} - \left\lfloor \frac{w'_v w'_u}{2W'_v}\right\rfloor\right)}\right] = E\left[\left(1-\frac{1}{w'_v}\right)^{B\left(\frac{w'_u w'_v}{2W'_u}\right)}\right]$$
$$= 1-\frac{w'_u w'_v}{2W'_u}+\frac{w'_u w'_v}{2W'_u}\left(1-\frac{1}{w'_v}\right) = 1-\frac{w'_u}{2W'_u} \leq e^{-\frac{w'_u}{2W'_u}}.$$

□

LEMMA 5.2. *Algorithm $\mathcal{A}$ computes a 2-approximate vertex cover.*

PROOF. The algorithm halts. In fact, the residual weight of each *active* node decreases by at least one in each round with positive probability. It follows that the nodes which do not switch to the *outside* state, switch to the *inside* state in a finite expected number of rounds.

Assume by contradiction that, at the end of the algorithm, the *inside* nodes do not form a vertex cover. This implies that there is an *outside* node v which has at least one *outside* neighbor. Let v switch to the state *outside* in phase p. At the beginning of phase $(p-1)$, all the neighbors of v are either *inside* or *active* nodes. Consider the *active* neighbors of v in phase $(p-1)$. These nodes are not active any more when phase p starts. But they cannot switch to the state *outside* in phase $(p-1)$, since their active degree is greater than zero in that phase. Thus they all switch to the state *inside*, which is a contradiction.

Let z_v be the difference between w_v and the final residual weight w'_v. A feasible solution of (D) is obtained by assigning to each dual variable y_{vu} the sum of all the counter-proposals of the kind $c_v(u)$ and $c_u(v)$. Let `apx` and `opt` be the weight of the vertex cover found and that of a minimum vertex cover, respectively. By weak duality:

$$apx \leq \sum_{v \in V} z_v \leq 2 \sum_{vu \in E} y_{vu} \leq 2\, opt.$$

Thus the vertex cover found is 2-approximate. □

LEMMA 5.3. *Algorithm $\mathcal{A}$ sends messages of size $O(\log W)$. Each phase of algorithm $\mathcal{A}$ has time complexity $O(\Delta \log(\Delta W))$ in expectation.*

PROOF. Both proposals and counter-proposals can be packed in messages of size $O(\log W)$.

The time complexity of each phase is upper bounded by the cost of computing the proposals. Computing the proposals is as expensive as selecting $O(\Delta)$ times an element out of $O(\Delta W)$ ones uniformly at random. Each random selection can be performed by generating $O(\log(\Delta W))$ random bits in expectation. By assuming a $O(1)$ cost for generating a random bit, the total expected cost of each phase is $O(\Delta \log(\Delta W))$. □

Recall that a node is *good* if at least one third of its neighbors have degree smaller or equal than its own degree. Consider a node v in G. The degree of all the micro-nodes corresponding to v in $\widetilde{G}$ is:

$$W_v = \sum_{u \in N(v)} w_u.$$

Thus a micro-node $v(i)$ is good if and only if:

$$\sum_{u \in N(v): W_u \leq W_v} w_u \geq \frac{W_v}{3}.$$

Note that, if a micro-node $v(i)$ is good, all the micro-nodes $v(j)$, $j \in \{1, 2 \dots w_v\}$, are good and vice-versa. We call a node of G *heavy* if all its micro-nodes in $\widetilde{G}$ are good. The next observation can be seen as the weighted analogue of Lemma 3.1.

LEMMA 5.4. *Let $E_H \subseteq E$ be the subset of edges incident to heavy nodes. Then:*

$$\sum_{vu \in E_H} w_v w_u \geq \frac{1}{2} \sum_{vu \in E} w_v w_u.$$

PROOF. Consider the auxiliary graph $\widetilde{G}$. The number of edges of $\widetilde{G}$ that are incident to good nodes is: $\sum_{\{v,u\} \in E_H} w_v w_u$. Since the number of edges of $\widetilde{G}$ is $\sum_{\{v,u\} \in E} w_v w_u$, the claim follows from Lemma 3.1. □

We use the properties of heavy nodes to prove the following bound on the number of rounds.

LEMMA 5.5. *Algorithm $\mathcal{A}$ halts in $O(\log n + \log \hat{W})$ expected rounds.*

PROOF. We show that the residual weight of heavy nodes decreases by at least a positive constant factor in expectation in each phase. It follows from Lemma 5.4 that the same holds for the potential function:

$$0 \leq \sum_{vu \in E} w'_v w'_u < (n\hat{W})^2,$$

thus implying the claim.

We condition on the values $\{w'_u\}_{u \in V}$ at the beginning of a given phase. Consider a heavy node v in that phase. Let w''_v be the value of w'_v at the end of the phase. The residual weight of v decreases by at least the sum of the counter-proposals $c_v(u)$ sent by v:

$$w''_v \leq w'_v - \sum_{u \in N'(v)} c_v(u) = w'_v - \min\{c'_v, \sum_{u \in N'(v)} p_u(v)\},$$

where, by definition, $w'_v = p'_v + c'_v$. Trivially,

$$\begin{aligned}
E[w''_v] &= Pr(c'_v = 0)E[w'_v | c'_v = 0] + Pr(c'_v \geq 1)E\left[w'_v - \sum_{u \in N'(v)} c_v(u) \,\middle|\, c'_v \geq 1\right] \\
&= Pr(c'_v = 0)E[p'_v | c'_v = 0] + Pr(c'_v \geq 1)E\left[p'_v + c'_v - \sum_{u \in N'(v)} c_v(u) \,\middle|\, c'_v \geq 1\right] \\
&= E[p'_v] + Pr(c'_v \geq 1)E\left[c'_v - \sum_{u \in N'(v)} c_v(u) \,\middle|\, c'_v \geq 1\right].
\end{aligned}$$

Now observe that, for $c'_v \geq 1$,

$$\begin{aligned}
c'_v - \sum_{u \in N'(v)} c_v(u) &= c'_v - \min\{c'_v, \sum_{u \in N'(v)} p_u(v)\} \\
&= c'_v \left(1 - \frac{\min\{c'_v, \sum_{u \in N'(v)} p_u(v)\}}{c'_v}\right) \\
&= c'_v \max\left\{0, 1 - \frac{\sum_{u \in N'(v)} p_u(v)}{c'_v}\right\} \\
&\leq c'_v \left(1 - \frac{1}{c'_v}\right)^{\sum_{u \in N'(v)} p_u(v)} \\
&\leq c'_v \prod_{u \in N'(v)} \left(1 - \frac{1}{w'_v}\right)^{p_u(v)}.
\end{aligned}$$

Recall that we condition over the values $\{w'_u\}_{u \in V}$. Under this condition, c'_v and

the $p_u(v)$'s are independent. By this observation and the inequality above:

$$E\left[c'_v - \sum_{u\in N'(v)} c_v(u) \,\middle|\, c'_v \geq 1\right] \leq E\left[c'_v \prod_{u\in N'(v)} \left(1-\frac{1}{w'_v}\right)^{p_u(v)} \,\middle|\, c'_v \geq 1\right]$$
$$= E[c'_v|c'_v \geq 1] \prod_{u\in N'(v)} E\left[\left(1-\frac{1}{w'_v}\right)^{p_u(v)} \,\middle|\, c'_v \geq 1\right]$$
$$= E\left[c'_v|c'_v \geq 1\right] \prod_{u\in N'(v)} E\left[\left(1-\frac{1}{w'_v}\right)^{p_u(v)}\right].$$

By Lemma 5.1 and the definition of heavy nodes:

$$\prod_{u\in N'(v)} E\left[\left(1-\frac{1}{w'_v}\right)^{p_u(v)}\right] \leq \prod_{u\in N'(v)} e^{-\frac{w'_u}{4W'_u}} \leq \prod_{u\in N'(v):W'_u\leq W'_v} e^{-\frac{w'_u}{4W'_v}} \leq e^{-\frac{1}{12}}.$$

We notice that $E[p'_v] = E[c'_v] = w'_v/2$. Moreover

$$E[c'_v] = Pr(c'_v = 0)E[c'_v|c'_v = 0] + Pr(c'_v \geq 1)E[c'_v|c'_v \geq 1]$$
$$= Pr(c'_v \geq 1)E[c'_v|c'_v \geq 1].$$

Altogether

$$E[w''_v] = E[p'_v] + Pr(c'_v \geq 1)E\left[c'_v - \sum_{u\in N'(v)} c_v(u) \,\middle|\, c'_v \geq 1\right]$$
$$\leq E[p'_v] + Pr(c'_v \geq 1)E[c'_v|c'_v \geq 1]e^{-\frac{1}{12}}$$
$$= E[p'_v] + E[c'_v]e^{-\frac{1}{12}} = \frac{w'_v}{2} + \frac{w'_v}{2}e^{-\frac{1}{12}} < w'_v.$$

□

Lemmas 5.2, 5.3, and 5.5 together imply Theorem 1.1.

REFERENCES

Bar-Yehuda, R. and Even, S. 1981. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms 2*, 198–203.

Chvátal, V. 1983. *Linear programming*. Freeman.

Cormen, T. H., Leiserson, C. E., and Rivest, R. L. 1992. *Introduction to algorithms*, 6th ed. MIT Press and McGraw-Hill Book Company.

Dinur, I. and Safra, S. 2002. The importance of being biased. In *ACM Symposium on the Theory of Computing (STOC)*. 33–42.

Garey, M. R. and Johnson, D. S. 1979. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. Freemann.

Halperin, E. 2002. Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. *SIAM Journal on Computing 31,* 5, 1608–1623.

Hańćkowiak, M., Karoński, M., and Panconesi, A. 2001. On the distributed complexity of computing maximal matchings. *SIAM Journal on Discrete Mathematics 15,* 1, 41–57.

Håstad, J. 2001. Some optimal inapproximability results. *Journal of the ACM 48,* 4, 798–859.

Hochbaum, D. S. 1982. Approximation algorithms for set covering and vertex cover problems. *SIAM Journal on Computing 11*, 555–556.

Israeli, A. and Itai, A. 1986. A fast and simple randomized parallel algorithm for maximal matching. *Information Processing Letters 22*, 77–80.

Karakostas, G. 2005. A better approximation ratio for the vertex cover problem. In *International Colloquium on Automata, Languages and Programming (ICALP)*. 1043–1050.

Khuller, S., Vishkin, U., and Young, N. 1994. A primal-dual parallel approximation technique applied to weighted set and vertex cover. *Journal of Algorithms 17,* 2, 280–289.

Monien, B. and Speckenmeyer, E. 1985. Ramsey numbers and an approximation algorithm for the vertex cover problem. *Acta Informatica 22*, 115–123.

Panconesi, A. and Rizzi, R. 2001. Some simple distributed algorithms for sparse networks. *DISTCOMP: Distributed Computing 14*.

Papadimitriou, C. and Yannakakis, M. 1991. Optimization, approximization and complexity classes. *Journal of Computer and System Sciences 43*, 425–440.

Received March 2006; revised December 2007; accepted May 2008.